Amanda Sumner
Kunskapskontroll 2
DS24 Deep Learning

Teoretiska frågor

1. Hur är AI, Maskininlärning och Deep Learning relaterat?

AI är det övergripande konceptet för maskiner som utför intelligenta uppgifter. Maskininlärning är ett sätt att uppnå AI där system lär sig från data. Deep Learning är en specialiserad typ av maskininlärning som använder komplexa neurala nätverk.

Kort sagt, deep learning är en del av maskininlärning, om i sin tur är en del av AI.

2. Hur är Tensorflow och Keras relaterat?

TensorFlow är ett bibliotek för storskalig maskininlärning. Keras är ett hög nivå anvärnarvänligt API för att bygga och träna neurala nätverk. Relationen är att Keras fungerar som ett enklare gränssnitt på TensorFlow. Keras är integrerat i den senaste version av TensorFlow och är det rekommenderade sättet att bygga modeller i TensorFlow.

3. Vad är en parameter? Vad är en hyperparameter?

Parameter är en variabel som modellen lär sig från träningsdatan. Parametrar används av modellen för att göra prediktioner. Deras värden bestäms baserat på datan, användaren sätter dem inte direkt.

Hyperparameter är en konfigurationsvariabel som ställs in manuellt av användaren eller hittas genom hyperparameter tuning innan träningsprocessen startar. De bestämmer hur modellen ska struktureras och tränas.

4. När man skall göra modellval och modellutvärdering kan man använda tränings-, validerings- och testdataset. Förklara hur de olika delarna kan användas.

Träningsdataset är den största delen av data och används för att träna modellen.

Valideringsdataset används fär modellval och hyperparameter justering. Efter att en modell har tränats på träningsdatasetet, utvärderas dess prestanda på valideringsdatasetet. Man kan testa olika värden för hyperparametrar, jämföra olika typer av modeller och välja de som ger bäst resultat på valideringsdatan.

Testdataset används endast en gång och ger en uppskattning av den slutliga, valda modellens faktiska prestanda på helt ny, osedd data, för att få en bedömning av hur väl modellen kommer att generalisera till ny data som den aldrig tidigare stött på.

5. Förklara vad nedanstående kod gör:

```
n_cols = x_train.shape[1]

nn_model = Sequential()
nn_model.add(Dense(100, activation='relu', input_shape=(n_cols, )))
nn_model.add(Dropout(rate=0.2))
nn_model.add(Dense(50, activation='relu'))
nn_model.add(Dense(1, activation='sigmoid'))

nn_model.compile(
    optimizer='adam',
    loss='binary_crossentropy',
    metrics=['accuracy' ])

early_stopping_monitor = EarlyStopping(patience=5)
nn_model.fit(
    x_train,
    y_train,
    validation_split=0.2,
    epochs=100,
    callbacks=[early_stopping_monitor])
```

`n_cols = x_train.shape[1]` hämtar antalet kolumner från träningsdatan `x_train`. `n_cols` kommer att användas för att definiera input lagrets storlek

`nn_model = Sequential()` initierar en sequential-modell som innebär att lagren i nätverket läggs till i en sekvens.

`nn_model.add(Dense(100, activation='relu', input_shape=(n_cols, )))` lägger till det första Dense lagret. I ett Dense lager varje neuron är anslutet till varje neuron i föregående lager. 100 är antalet neuroner i detta lager. `activation='relu'` anger aktiveringsfunktion ReLU (Rectified Linear Unit). `input_shape=(n_cols, )` definierar formen på indatan för den första lagret. Kommatecknet efter `n_cols` indikerar att et är en tuple, även om det bara är en dimension.

`nn_model.add(Dropout(rate=0.2))` lägger till ett Dropout lager, en regulariseringsteknik för att förhindra overfitting. `rate=0.2` betyder att under träningen kommer 20% av neuronerna från det föregående lagret slumpmässigt att stängas av vid varje uppdateringssteg.

`nn_model.add(Dense(50, activation='relu'))` lägger till ett andra Dense lager, 50 är antalet neuroner i detta lager, använder ReLU som aktiveringsfunktion.

`nn_model.add(Dense(1, activation='sigmoid'))` lägger till ett output lager med en enda neuron i utdata. Detta används för binär klassificering där modellen ska förutsäga en av två klasser. `activation='sigmoid'` producerar ett värde mellan 0 och 1, vilket tolkas som sannolikheten får att indatan tillhör den positiva klassen.

```
nn_model.compile(
      optimizer='adam',
      loss='binary_crossentropy',
      metrics=['accuracy' ])
```
konfigurerar inlärningsprocessen för modellen innan träning. Optimeringsalgoritmen `'adam'` anpassar inlärningstakten under träning. `loss='binary_crossentropy'` anger loss `function. metrics=['accuracy']` anger vilket mätvärden som ska användas för att utvärdera modellen under träning och testing.

`early_stopping_monitor = EarlyStopping(patience=5)` initierar en EarlyStopping callback som förhinder överanpassning genom att avsluta träningen i förtid. `patience=5` avbrytar träningen om det övervakade mätverdet inte har förbättrats under de senaste 5 epochs.

```
nn_model.fit(
      X_train,
      Y_train,
      validation_split=0.2,
      epochs=100,
      callbacks=[early_stopping_monitor])
```
tränar modellen på den angivna datan. `x_train` och `y_train` är träningsdatan. `validation_split=0.2` anger att 20% av träningsdatan ska avsättas som valideringsdata. `epochs=100` är det maximala antalet gånger modellen kommer att iterera över träningsdatasetet om träningen har inte avslutats av EarlyStopping. `callbacks=[early_stopping_monitor]` är en lista med callbacks som ska anropas under träningen, i detta fall är det bara den tidigare definierade `early_stopping_monitor`.

6. Vad är syftet med att regularisera en modell?

Syftet med att regularisera en modell är att förhindra overfitting och förbättra modellens generaliseringsförmåga på ny data den inte har sett tidigare. Regularisering kan hantera kollinearitet och funktionsurval.

7. ”Dropout” är en regulariseringsteknik, vad är det för något?

Dropout används för att motverka overfitting. Under träning, vid varje träningsiteration stängs av en slumpmässigt vald andel neuroner i ett lager. Andelen neuroner som stängs av eller dropout rate, t.ex. 0.2 som i föregående kodexempel, är en hyperparameter som ställs in innan träningen.

8. ”Early stopping” är en regulariseringsteknik, vad är det för något?

Early stopping är en form av regularisering som också används för att förhindra overfitting. Modellen tränas i flera epochs, i varje epoch justeras modellens parametrar. Ett specifikt modellens prestandamått utvärderas efter varje epoch på ett separat valideringsdataset, t.ex., accuracy. Träningen avbryts när prestandan slutar förbättras eller börjar försämras. Early stopping minskar träningstid och resursanvändning.

9. Din kollega frågar dig vilken typ av neuralt nätverk som är populärt för bildanalys, vad svarar du?

Den vanligaste typen av neuralt nätverk för bildanalys är Convolutional Neural Networks (CNN). Ett nyare alternativ till CNN är Vision Transformers (ViTs) som använder NLP arkitektur ursprungligen utvecklad fär språkbehandling. Men CNN är det mest kända och brett använda nätverket för bildanalys.

10. Förklara översiktligt hur ett ”Convolutional Neural Network” fungerar.

Ett CNN fungerar genom att hierarkistk lära sig känna igen mönster och strukturer i bilden. Bilden matas in som en matris av pixelvärden, med tre matriser för färgbilder för RGB värden. CNN detekterar features med convolutional layers, som använder filters över hela bilden och utför en matematisk operation mellan filtrets vikter och en del av bilden. Resultat från varje filter är en feature map som visar var i bilden den specifika feature finns. Sedan pooling layers reducerar storleken på feature maps och minskar parametrar och beräkningar i nätverket. Utdata från den sista lager sättas in till en endimensionell vektor. Denna vektor matas in i ett vanligt neural netvärkslager som används för att utföra den slutliga klassificeringen.

11. Vad gör nedanstående kod?

```
model.save("model_file.keras")
my_model = load_model("model_file.keras")
```

Koden sparar och laddar en tränad maskininlärningsmodell med Keras biblioteket. `.keras` formatet är det rekommenderade sättet att spara Keras-modeller. Den laddade modellen tilldelas till en ny variabel `my_model` som är en kopia av den modell som ursprungligen sparades och can användas för att göra prediktioner eller fortsätta tränas.

12. Deep Learning modeller kan ta lång tid att träna, då kan GPU via t.ex. Google Colab skynda på träningen avsevärt. Skriv mycket kortfattat vad CPU och GPU är.

CPU är central processing unit, den primära komponenten av en dator som utförar instruktioner och beräkningar sekventiellt med ett fåtal kärnor. GPU är graphics processing unit, urpsungligen för grafik, men det har tusentals mindre kärnor som är bra att använda för att utföra många beräkningar parallelt, vilket snabbar upp maskininlärning.